達人プログラマーの感想

安彦 久志

2021/02/22

1　はじめに

私は今回初めて，ソフトウェア開発全般に関わる書籍を読んだ．全体の印象としては，比喩が独特で理解出来ない箇所もあったが，比喩のおかげで記憶に残る内容であった．今回は，当書において特に気になった内容について，私が大学院の研究のために行ったソフトウェア開発に当てはめて考えた．

2　割れた窓を放置しておかないこと

1 枚の割れた窓（悪い設計や質の悪いコード）を放置することで，ソフトウェア全体の質が悪くなってしまうという内容が述べられていた．私の場合，質の悪いコードがあった場合，簡単な対処をすることが多かった．例えば，どういった処理が悪いのかコメントを残したり，バグが疑われる箇所や修正すべき箇所をリストアップしておく等の対処をしていた．しかし，設計については対処できておらず，設計の良し悪しを判断することも出来ていなかったと思う．とはいえ，設計に対する知見はまだまだ浅いため，仕事の中で広げていきたいと思った．

3　DRY–繰り返しを避けること

ソフトウェアの信頼性と保守性が高くするためには繰り返しを避けなければならないという内容が述べられていた．私の場合，手抜きのために二重化をしてしまうことが頻繁にあった．例えば，パラメータの設定を 2 つのコードに直書きしていることがあった．その数値は変更しない前提で開発していたが，数値に誤りがあり，修正するときに一方のコードしか修正しておらず，期限ギリギリになってバグが見つかり，この二重化に気づくのに時間が掛かったことがあった．仕事では，私が二重化を犯してしまうことで，プロジェクト全体の信頼性，保守性に影響してしまう可能性もあるため，二重化には十分注意したい．

4　モジュール間の結合度を最小にする

モジュール間のやりとりを最小限にすることで，変更が起きたときの影響範囲を小さくするという内容が述べられていた．私の場合，言語の選択肢として C と C++があったが，クラス設計が出来ず，C 言語で開発を行った．また，主要な構造体をグローバル変数で定義し，ほとんどの関数でグローバル変数を用いることとなった．結果として，保守性が著しく低く，機能追加の際には多くの作業を要してしまった．まずは，オブジェクト指向の設計を理解するところから始めた．

5　テスト設計を行うこと

開発の初期の段階でテスト機構を構築し，ことあるごとにテストを実行すべきという内容が述べれていた．私の場合，開発の中でテストは注意深く行っており，多くのテストをコード内に実装していた．これにより，多くのバグを発見することができ，非常に役立った．しかし，全ての機能をテストすることは出来ておらず，テストコードを組み込む箇所は感覚に頼っていた．これに対して，当書ではコードの実装より前にテストを作成することを考えている．期限に追われるなかでテストを先に作成することは，現実的ではないようにも感じた．しかし，テストを作成することは，ソフトウェアの出力について，制約条件を確認することになるので，システムを俯瞰して考えることができると感じた．

6　ドキュメントは付け足すものでなく，組み込むものである

コード内に適切なコメントを記述し，コメントからドキュメントを作成するツールを用いることで，容易にドキュメントを作成できるという内容が述べれていた．私は，これまでドキュメントを書いたことはなく，コーディングとは別の作業だと想像していた．しかし，コメントに基づくドキュメント作成はメリットが多く，非常に便利だと感じた．また，ドキュメントはコードに対する単なるビューとなるという考えは，プログラマらしく面白いと思った．

7　おわりに

当書は，経験の乏しい私でも理解できる内容が多かった，しかし，まだまだ理解できない箇所も多く，広い層のエンジニアが読めるような奥深い内容だと感じた．大学の研究でも，学部 4 年生の頃から何度も読み返している書籍があり，当書にも似たものを感じた．これから，エンジニアとして経験を積みながら，当書を読み返し，達人プログラマになれるよう精進していきたい．